Muhammad Abdullah

STU-DS-251-930

Heart Disease Prediction Project

Project Overview

This project aims to build a predictive model for heart disease diagnosis using machine learning. We use a dataset that contains health-related attributes such as age, sex, chest pain type, cholesterol level, resting blood pressure, etc. The goal is to classify whether a person is likely to have heart disease (1) or not (0), based on these features.

1. Importing Libraries

We begin by importing the necessary Python libraries required for data handling, visualization, and machine learning tasks:

- `numpy` and `pandas` for data manipulation
- `matplotlib.pyplot` and `seaborn` for plotting
- `sklearn` for model training, evaluation, and preprocessing

2. Load Dataset

The dataset (`heart.csv`) is loaded using `pandas`. We perform an initial inspection using `.head()` and `.info()` to understand the number of features, data types, and presence of any missing values.

3. Data Cleaning

We ensure data integrity by:

- Checking for and removing missing values (if any)
- Dropping duplicate rows to prevent model bias
  This step is essential for ensuring accurate model training.

4. Categorical Data Handling

Some columns like `cp`, `thal`, and `slope` are categorical. Since machine learning models work better with numerical inputs, we convert these columns into multiple binary (0/1) columns using **one-hot encoding**. This helps preserve the categorical nature while making it model-friendly.

5. Feature Scaling

Feature scaling is applied using **StandardScaler**, which standardizes values to a mean of 0 and standard deviation of 1. This ensures that features like cholesterol and age—despite having different units—are treated equally by the model.

6. Train-Test Split

To evaluate model performance fairly, we split our dataset into:

- **Training Set (80%)** – used to train the machine learning model.
- **Testing Set (20%)** – used to test how well the model generalizes to new data.

We use `train_test_split()` from scikit-learn to perform this split randomly.

7. Exploratory Data Analysis (EDA)

In this section, we explore relationships between features and the target using visual tools:

- A **countplot** shows the distribution of heart disease (target variable).
- A **heatmap** displays correlations between all features, helping us identify important variables.
- A **boxplot** is used to compare the distribution of age between patients with and without heart disease.

These visualizations help us understand which features may influence the prediction.

8. Feature Selection

Using the correlation matrix, we examine how strongly each feature is correlated with the target. Features with high correlation are likely more useful in model training, and this insight guides our model building.

9. Model Training & Evaluation

We trained two models in this project:

Decision Tree Classifier

- A non-linear model that splits data into decision branches.
- Easy to interpret but can overfit if not tuned properly.

Support Vector Machine (SVM)

- A linear classifier that tries to find the best margin (hyperplane) to separate classes.
- Works well with scaled data and high-dimensional features.

For both models, we:

- Trained them on the training dataset.
- Made predictions on the test set.
- Evaluated performance using:
    - **Accuracy** – overall correctness
    - **Precision** – how many predicted positives are correct
    - **Recall** – how many actual positives were identified
    - **F1 Score** – harmonic mean of precision and recall
    - **Confusion Matrix** – detailed count of true/false positives/negatives

10. Model Comparison

We created a comparison table with metrics for both models side-by-side. This helps us decide which model performs better and is more suitable for real-world application.

Conclusion

The notebook successfully demonstrates a machine learning pipeline—from loading and cleaning data to training and evaluating multiple models. By comparing model metrics, we can select the most reliable model for heart disease prediction. Future work could involve hyperparameter tuning, feature engineering, and testing other algorithms like Random Forest or Logistic Regression for improved accuracy.